

```
EEEEEEEEEE XX XX AAAAAA MM MM PPPPPPPP LL EEEEEEEEEE SSSSSSSS
EEEEEEEEEE XX XX AAAAAA MM MM PPPPPPPP LL EEEEEEEEEE SSSSSSSS
EEEEEEEEEE XX XX AAAAAA MM MM PPPPPPPP LL EEEEEEEEEE SSSSSSSS
EE XX XX AA AA MMMM MMMM PP PP LL EE SS
EE XX XX AA AA MMMM MMMM PP PP LL EE SS
EE XX XX AA AA MM MM PP PP LL EE SS
EE XX XX AA AA MM MM PP PP LL EE SS
EE XX XX AA AA MM MM PP PP LL EE SS
EEEEEEEEEE XX XX AA AA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSS
EEEEEEEEEE XX XX AA AA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSS
EEEEEEEEEE XX XX AA AA MM MM PPPPPPPP LL EEEEEEEEE SSSSSSS
EE XX XX AAAAAAAAAA MM MM PP LL EE SS
EE XX XX AAAAAAAAAA MM MM PP LL EE SS
EE XX XX AAAAAAAAAA MM MM PP LL EE SS
EE XX XX AA AA MM MM PP LL EE SS
EE XX XX AA AA MM MM PP LL EE SS
EEEEEEEEEE XX XX AA AA MM MM PP LLLLLLLLLL EEEEEEEEE SSSSSSS
EEEEEEEEEE XX XX AA AA MM MM PP LLLLLLLLLL EEEEEEEEE SSSSSSS
EEEEEEEEEE XX XX AA AA MM MM PP LLLLLLLLLL EEEEEEEEE SSSSSSS
```

```
DDDDDDDD 000000 DDDDDDDDD
DDDDDDDD 000000 DDDDDDDDD
DD DD DD 00 00 DD DD
DD DD DD 00 00 DD DD
DD DD DD 00 00 DD DD
DD DD DD 00 00 DD DD
DD DD DD 00 00 DD DD
DD DD DD 00 00 DD DD
DD DD DD 00 00 DD DD
DD DD DD 00 00 DD DD
DD DD DD 00 00 DD DD
DDDDDDDD 000000 DDDDDDDDD
DDDDDDDD 000000 DDDDDDDDD
-----
EEEEEEEEEE RRRRRRRR AAAAAA PPPPPPPP AAAAAA TTTTTTTTTT
EEEEEEEEEE RRRRRRRR AAAAAA PPPPPPPP AAAAAA TTTTTTTTTT
EE RR RR AA AA PP PP AA AA TT
EE RR RR AA AA PP PP AA AA TT
EE RR RR AA AA PP PP AA AA TT
EEEEEEEEEE RRRRRRRR AA AA PPPPPPPP AA AA TT
EEEEEEEEEE RRRRRRRR AA AA PPPPPPPP AA AA TT
EE RR RR AAAAAAAAAA PP AAAAAAAAAA TT
EE RR RR AAAAAAAAAA PP AAAAAAAAAA TT
EE RR RR AA AA PP AA AA TT
EE RR RR AA AA PP AA AA TT
EEEEEEEEEE RRRRRRRR AA AA PP AA AA TT
EEEEEEEEEE RRRRRRRR AA AA PP AA AA TT
```

```
MM MM AAAAAA RRRRRRRR
MM MM AAAAAA RRRRRRRR
MMMM MMM AA AA RR RR
MMMM MMM AA AA RR RR
MM MM AA AA RR RR
MM MM AA AA RRRRRRRR
MM MM AA AA RRRRRRRR
MM MM AAAAAAAAAA RR RR
MM MM AAAAAAAAAA RR RR
MM MM AA AA RR RR
MM MM AA AA RR RR
MM MM AA AA RR RR
MM MM AA AA RR RR
```

.title DOD\_ERAPAT - Generate DoD security erase patterns  
.ident 'V03-000'

\*\*\*\*\*  
\* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY \*  
\* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. \*  
\* ALL RIGHTS RESERVED. \*  
\*\*\*\*\*

\*\*\*\*\*  
\* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED \*  
\* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE \*  
\* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER \*  
\* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY \*  
\* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY \*  
\* TRANSFERRED. \*  
\*\*\*\*\*

\*\*\*\*\*  
\* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE \*  
\* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT \*  
\* CORPORATION. \*  
\*\*\*\*\*

\*\*\*\*\*  
\* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS \*  
\* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. \*  
\*\*\*\*\*

++  
Facility:

VMS Executive

Abstract:

This routine generates security erase patterns which are used by user  
written programs to preclude the unauthorized disclosure of classified  
information.

Environment:

VAX/VMS, Kernel Mode

Author:

Michael T. Rhodes, Creation Date: October, 1982

Modified By:

V03-001 JRL0023 John R. Lawson, Jr. 10-Jul-1984 14:23  
Add interface to the system.

--  
.page



.sbtll Declarations

\$ERADF                   : Define function codes  
 \$\$\$DEF                 : Define status codes

: Equated symbols:

TYPE     = 4                 : Offset to TYPE parameter (value)  
 COUNT   = 8                 : Offset to COUNT parameter (value)  
 PATADR   = 12                : Offset to PATADR parameter (address)

: Assumptions:

ASSUME ERASK\_MINTYPE EQ 1  
 ASSUME ERASK\_MAXTYPE EQ 3

ASSUME ERASK\_MEMORY EQ 1  
 ASSUME ERASK\_DISK EQ 2  
 ASSUME ERASK\_TAPE EQ 3

.page  
 .sbtll Loadable image header and trailer

:++

: Loader Information:

At boot time, SYSBOOT.EXE checks the SYSGEN parameter LOADERAPT  
 (SGNSV LOADERAPT); if it is set, this image gets loaded from  
 SYSSYSTEM:ERAPATLOA.EXE. There must exist, in the image, certain  
 information for the loader; these two PSECT's supply that info.

: Linking this Object:

\$ link/notraceback/system=0/header/executable=SYSSYSTEM:ERAPATLOA -  
 DOD\_ERAPAT, SYSSYSTEM:SYS.STB/selective\_search

The /SYSTEM qualifier guarantees that the PSECT's will be ordered  
 alphabetically within the image, forcing \$\$\$\$\$\$\$ to be first and  
 ----- to be last.

:--

: This table must appear at the beginning of the image

.psect \$\$\$\$\$\$\$\$           page, pic         : In a system image, the PSECT's  
   : are ordered alphabetically  
 PRMSW = 1                 : Flag to indicate loadable code

:

; This table directs SYSBOOT.EXE for loading the rest of this image

```
SLVTAB  subtype=DYN$C NON PAGED,-      ; Non-paged pool
        end=DOD_ERAPAT$END,-          ; Computed size of image
        sysvecs=VECTOR$$,-            ; Vector into the routine
        prot_w=PRT$C URKW,-           ; Page protection
        facility=<DoD Security Erase> ; What is this?
```

; These vectors replace the default ones in SYS.EXE

```
VECTOR$$:                                ; Vector table
        LOADVEC EXE$ERAPAT_VEC,5,,DOD_ERAPAT$
        .long  -1                        ; Terminated by -1
```

; This label must appear at the end of the image

```
.psect  -----      byte, pic
```

DOD\_ERAPAT\$END::

```
.page
.sbttl  $ERAPAT System Service
```

++  
\$ERAPAT

#### Functional Description:

To preclude the unauthorized disclosure of classified information, the caller iteratively invokes the \$ERAPAT system service. Upon each invocation, the user increments the iteration count and the service returns an erasure pattern plus either SSS\_NORMAL or SSS\_NOTRAN (which indicates the declassification procedure is complete).

#### Calling sequence:

This routine should be called via a CALLS/G to EXE\$ERAPAT.

#### Input:

```
TYPE(AP)      Security erase type. The legal types are
               1. ERASK_MEMORY : main memory
               (volatile r/w semiconductor)
               2. ERASK_DISK   : disk storage
               3. ERASK_TAPE   : tape storage

COUNT(AP)     Iteration count. The service should be called
```

the first time with the value 1, then 2, etc., until the status \$\$\$NOTRAN is returned. The local symbol MAXCOUNT defines how many times this happens.

## Output:

PATADR(AP) Address of a longword into which the security erase pattern is to be written.

## Routine value:

RO = \$\$\$ACCVIO	Pattern output area not accessible
\$\$\$BADPARAM	Invalid security type code
\$\$\$NORMAL	Normal successful completion
\$\$\$NOTRAN	Security erase complete

--

.page  
 .sbttl Data necessary for routine  
 .psect \$DATA\$ long, pic

## Own Storage:

## COUNTS\$:

.long 1	; Main Memory iteration count
.long 3	; Disk Storage iteration count
.long 2	; Tape Storage iteration count

## PATTERNS\$:

.long 0	; Main memory erase pattern
.long -1	; Disk Storage erase pattern
.long ^XDB6DB6DB	; Tape Storage erase pattern

.page  
 .sbttl Routine to generate the erase patterns  
 .psect \$CODE\$ long, pic

## Routine to return erase patterns:

## DOD\_ERAPAT\$:

pushr ^M<r1> ; Save registers

## Check the values of the parameters ...



```

;
    movzwl  #SS$_BADPARAM, r0      ; Assume bad parameters
    cmpl    COUNT(ap), #0          ; Is count too small?
    bleq    EXIT                  ; Branch if yes

    cmpl    TYPE(ap), #ERASK_MINTYPE ; Type code too small?
    blss    EXIT                  ; Branch if yes
    cmpl    TYPE(ap), #ERASK_MAXTYPE ; Type code too large?
    bgtr    EXIT                  ; Branch if yes

; Use the TYPE as an index into COUNTSS and PATTERNSS
;
    subl3   #1, TYPE(ap), r1       ; Vectors begin with 0

; Signal completion by returning SS$_NOTRAN ...
;
    movzwl  #SS$_NOTRAN, r0        ; Set completion status
    cmpl    COUNT(ap), COUNTSS[r1] ; Are we done?
    bgtr    EXIT                  ; Yes, return completion status

; Is the return address for the pattern writable ???
;
    movzwl  #SS$_ACCVIO, r0        ; Assume access violation
    ifnowrt #4, @PATADR(ap), EXIT  ; Branch if no write access

; Look up the appropriate erase pattern ...
;
    movzwl  #SS$_NORMAL, r0        ; Assume success at this point
    movl    PATTERNSS[r1], @PATADR(ap) ; Send back the pattern

; That's all folks ...
;
EXIT:  popr  ^M<r1>                ; Restore registers
      ret                                ; Return
      .END

```



0157 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY